

# Commands SHELL

---

## bash

**Description:** bash is a command language interpreter. It is widely available on various operating systems and is a default command interpreter on most GNU/Linux systems. The name is an acronym for the 'Bourne-Again SHell'.

### Options:

- **-c** string: Read and execute the commands in the string argument.
- **-i**: Start an interactive shell session.
- **-l**: Make bash act as if it had been invoked as a login shell.
- **-r**: Make bash act as if it had been invoked as a restricted shell.
- **-s**: Read commands from the standard input.
- **-D**: A list of all double-quoted strings preceded by \$ is printed on the standard output.

```
bash
```

## cat

**Description:** cat is a standard Unix utility that reads files sequentially, writing them to standard output. The name is derived from its function to concatenate files.

### Options:

- **-A, --show-all**: Equivalent to -vET.
- **-b, --number-nonblank**: Number all nonempty output lines, starting with 1.
- **-e**: Equivalent to -vE.
- **-E, --show-ends**: Display a \$ after the end of each line.
- **-n, --number**: Number all output lines, starting with 1.
- **-s, --squeeze-blank**: Suppress repeated empty output lines.
- **-t**: Equivalent to -vT.
- **-T, --show-tabs**: Display TAB characters as ^I.
- **-u**: Disable output buffering.
- **-v, --show-nonprinting**: Display control characters.

```
cat
```

## cd

**Description:** cd is a command-line utility for changing the current working directory in operating systems.

### Options:

- **-L**: Force symbolic links to be followed.
- **-P**: Use the physical directory structure without following symbolic links.

```
cd
```

# chmod

**Description:** chmod is a command in Unix and Unix-like operating systems that allows to change the file mode bits of files and directories.

## Options:

- **-c, --changes**: Like verbose but report only when a change is made.
- **-f, --silent, --quiet**: Suppress most error messages.
- **-v, --verbose**: Output a diagnostic for every file processed.
- **-R, --recursive**: Change files and directories recursively.

## octal mode:

- **0**: No permission.
- **1**: Execute permission.
- **2**: Write permission.
- **3**: Write and execute permissions.
- **4**: Read permission.
- **5**: Read and execute permissions.
- **6**: Read and write permissions.
- **7**: Read, write, and execute permissions.

user: **u**, group: **g**, others: **o**, all: **a**. calculate the octal mode by adding the permissions for each user type.

## example:

- **chmod 700 file.txt** -> **user**: read, write, execute; **group**: no permission; **others**: no permission.
- **chmod 755 file.txt** -> **user**: read, write, execute; **group**: read, execute; **others**: read, execute.
- **chmod 644 file.txt** -> **user**: read, write; **group**: read; **others**: read.

## example of using the symbolic mode:

- **chmod u+x file.txt** -> add execute permission to the user.
- **chmod g-w file.txt** -> remove write permission from the group.
- **chmod o=r file.txt** -> set read permission to others.
- **chmod a=rwx file.txt** -> set read, write, and execute permissions to all.

# cp

**Description:** cp is a command-line utility for copying files and directories.

## Options:

- **-a, --archive**: Preserve as much as possible of the structure and attributes of the original files in the copy.
- **-b, --backup**: Make a backup of each existing destination file.
- **-f, --force**: If an existing destination file cannot be opened, remove it and try again.
- **-i, --interactive**: Prompt whether to overwrite an existing destination file.
- **-R, --recursive**: Copy directories recursively.
- **-p, --preserve**: Preserve the specified attributes of the original files.
- **-u, --update**: Copy only when the source file is newer than the destination file or when the destination file is missing.
- **-v, --verbose**: Print the name of each file before copying it.

```
cp
```

## date

**Description:** date is a command-line utility that displays the current date and time.

### Options:

- **-d, --date**: Display the date and time specified in the argument.
- **-u, --utc**: Display the date and time in Coordinated Universal Time (UTC).
- **-R, --rfc-2822**: Display the date and time in RFC 2822 format.
- **-I, --iso-8601**: Display the date and time in ISO 8601 format.
- **-r, --reference**: Display the date and time of the file specified in the argument.
- **-s, --set**: Set the date and time to the value specified in the argument.

```
date
```

## diff

**Description:** diff is a command-line utility for comparing files line by line.

### Options:

- **-c, --context**: Output three lines of context around each difference.
- **-u, --unified**: Output three lines of unified context around each difference.
- **-n, --rcs**: Output an RCS format diff.
- **-y, --side-by-side**: Output in two columns.
- **-W, --width**: Output at most n (default 130) print columns.
- **-l, --paginate**: Pass the output through pr to paginate it.
- **-i, --ignore-case**: Ignore case differences in file contents.
- **-b, --ignore-space-change**: Ignore changes in the amount of white space.
- **-B, --ignore-blank-lines**: Ignore changes whose lines are all blank.
- **-a, --text**: Treat all files as text and compare them line by line.
- **-q, --brief**: Output only whether files differ.
- **-s, --report-identical-files**: Output only whether files are identical.
- **-r, --recursive**: Recursively compare any subdirectories found.
- **-N, --new-file**: Treat absent files as empty.
- **-d, --minimal**: Try to find a smaller set of changes.
- **--normal**: Output a normal diff.

```
diff
```

## echo

**Description:** echo is a command-line utility that prints the specified text to the standard output.

### Options:

- **-n**: Do not output the trailing newline.

- **-e**: Enable interpretation of backslash escapes.

```
echo
```

## env

**Description:** env is a command-line utility that runs another command with a modified environment.

### Options:

- **-i**: Clear the environment before running the command.
- **-u**: Remove variable from the environment.

```
env
```

## exit

**Description:** exit is a command-line utility that terminates the shell.

```
exit
```

## export

**Description:** export is a command-line utility that sets an environment variable.

```
export
```

## grep

**Description:** grep is a command-line utility for searching plain-text data sets for lines that match a regular expression.

### Options:

- **-c, --count**: Suppress normal output; instead, print a count of matching lines for each input file.
- **-i, --ignore-case**: Ignore case distinctions in both the PATTERN and the input files.
- **-v, --invert-match**: Invert the sense of matching, to select non-matching lines.
- **-w, --word-regexp**: Select only those lines containing matches that form whole words.
- **-x, --line-regexp**: Select only those matches that exactly match the whole line.
- **-f, --file**: Obtain patterns from FILE, one per line.
- **-r, --recursive**: Read all files under each directory, recursively.
- **-n, --line-number**: Prefix each line of output with the 1-based line number within its input file.
- **-h, --no-filename**: Suppress the prefixing of file names on output.
- **-o, --only-matching**: Print only the matched (non-empty) parts of a matching line, with each such part on a separate output line.
- **-q, --quiet, --silent**: Quiet; do not write anything to standard output.
- **-s, --no-messages**: Suppress error messages about nonexistent or unreadable files.

- `-z, --null-data`: Treat input and output data as sequences of lines, each terminated by a zero byte (the ASCII NUL character) instead of a newline.
- `--binary-files`: Determine how to deal with binary files.
- `--text`: Treat all files as text.

### speciale characters:

- `.`: Any single character.
- `^`: The beginning of a line.
- `$`: The end of a line.
- `[]`: Any one of the characters inside the square brackets.
- `[^]`: Any character that is not inside the square brackets.
- `*`: Zero or more of the preceding character.
- `+`: One or more of the preceding character.
- `?`: Zero or one of the preceding character.
- `|`: Alternation.
- `()`: Grouping.
- `\`: Escape the next character.
- `{}`: Specify the number of occurrences of the preceding character.

### examples of special characters:

- `grep '^a' file.txt`: lines that start with the letter 'a'.
- `grep 'a$' file.txt`: lines that end with the letter 'a'.
- `grep 'a.' file.txt`: lines that have the letter 'a' followed by any character.
- `grep 'a*' file.txt`: lines that have the letter 'a' followed by zero or more characters.
- `grep 'a+' file.txt`: lines that have the letter 'a' followed by one or more characters.
- `grep 'a?' file.txt`: lines that have the letter 'a' followed by zero or one character.
- `grep 'a|b' file.txt`: lines that have the letter 'a' or the letter 'b'.
- `grep 'a(b|c)' file.txt`: lines that have the letter 'a' followed by the letter 'b' or the letter 'c'.
- `grep 'a{2}' file.txt`: lines that have the letter 'a' repeated two times.
- `grep 'a{2,4}' file.txt`: lines that have the letter 'a' repeated from two to four times.
- `grep 'a{2,}' file.txt`: lines that have the letter 'a' repeated two or more times.
- `grep 'a{,4}' file.txt`: lines that have the letter 'a' repeated up to four times.
- `grep 'a[bc]' file.txt`: lines that have the letter 'a' followed by the letter 'b' or the letter 'c'.
- `grep 'a[^bc]' file.txt`: lines that have the letter 'a' followed by any character that is not the letter 'b' or the letter 'c'.
- `grep 'a[b-d]' file.txt`: lines that have the letter 'a' followed by the letter 'b', 'c', or 'd'.
- `grep 'a[^b-d]' file.txt`: lines that have the letter 'a' followed by any character that is not the letter 'b', 'c', or 'd'.
- `grep 'a[a-z]' file.txt`: lines that have the letter 'a' followed by any lowercase letter.
- `grep 'a[a-zA-Z]' file.txt`: lines that have the letter 'a' followed by any letter.
- `grep 'a[0-9]' file.txt`: lines that have the letter 'a' followed by any digit.
- `grep 'a[0-9a-f]' file.txt`: lines that have the letter 'a' followed by any digit or the letter 'a', 'b', 'c', 'd', 'e', or 'f'.

grep

## head

**Description:** head is a command-line utility that prints the first N lines of a file.

### Options:

- `-n`, `--lines`: Print the first N lines of each file.

speical characters:

- `+N`: Start displaying the file from the Nth line.

examples:

- `head -n 5 file.txt`: print the first 5 lines of the file.
- `head -n -5 file.txt`: print all lines except the last 5 lines of the file.
- `head -n +5 file.txt`: print all lines starting from the 5th line of the file.

```
head
```

## id

**Description:** id is a command-line utility that prints the user and group IDs of the current user.

```
id
```

## kill

**Description:** kill is a command-line utility that sends a signal to a process.

### Options:

- `-l`, `--list`: List the signal names.
- `-s`, `--signal`: Specify the signal to send.

```
kill
```

## ln

**Description:** ln is a command-line utility that creates a link to a file.

### Options:

- `-s`, `--symbolic`: Create a symbolic link.
- `-f`, `--force`: Remove existing destination files.
- `-i`, `--interactive`: Prompt whether to remove existing destination files.
- `-n`, `--no-dereference`: Treat destination files as a normal file if they are symbolic links.
- `-v`, `--verbose`: Print the name of each file before linking it.

```
ln
```

## ls

**Description:** ls is a command-line utility that lists files and directories.

### Options:

- **-a, --all**: Include directory entries whose names begin with a dot.
- **-A, --almost-all**: Include directory entries whose names begin with a dot, except for the . and .. entries.
- **-c**: Use time of last modification of the file status information for sorting.
- **-C**: List entries by columns.
- **-d, --directory**: List directory entries instead of contents.
- **-f**: Do not sort; list entries in directory order.
- **-g**: Like -l, but do not list owner.
- **-h, --human-readable**: Print sizes in human-readable format.
- **-i, --inode**: Print the index number of each file.
- **-l**: List in long format.
- **-m**: List entries by rows.
- **-n**: List numeric user and group IDs.
- **-q, --hide-control-chars**: Print ? instead of nongraphic characters.
- **-Q, --quote-name**: Quote file names with double quotes.
- **-r, --reverse**: Reverse order while sorting.
- **-R, --recursive**: List subdirectories recursively.
- **-s, --size**: Print the allocated size of each file, in blocks.
- **-S**: Sort by file size.
- **-t**: Sort by modification time.
- **-u**: Use time of last access for sorting.
- **-v**: Sort by version.
- **-w, --width**: Set screen width to n columns.
- **-X**: List entries by lines instead of by columns.
- **-X**: Sort alphabetically by entry extension.
- **-1**: List one file per line.

```
ls
```

## man

**Description** man is a command-line utility that displays the manual pages.

### Options:

- **-k, --apropos**: Search the manual page names and descriptions.
- **-f, --whatis**: Search the manual page names.
- **-w, --where**: Show the locations of the manual pages.
- **-l, --local-file**: Show the location of the manual page source file.
- **-S, --sections**: List the manual page sections.
- **-7**: Display the manual page in ASCII format.

```
man
```

## mkdir

**Description:** mkdir is a command-line utility that creates directories.

**Options:**

- **-m, --mode:** Set the file mode.
- **-p, --parents:** Create parent directories as needed.
- **-v, --verbose:** Print a message for each created directory.

```
mkdir
```

## more

**Description:** more is a command-line utility that displays the contents of a file one screen at a time.

**Options:**

- **-d:** Display help.
- **-f:** Count logical rather than screen lines.
- **-l:** Ignore form feeds.
- **-p:** Do not scroll.
- **-c:** Do not scroll.
- **-s:** Squeeze multiple blank lines.
- **-u:** Suppress underlining.
- **-n:** Specify the number of lines per screen.

```
more
```

## mv

**Description:** mv is a command-line utility that moves files and directories. It can also rename files and directories.

**Options:**

- **-b, --backup:** Make a backup of each existing destination file.
- **-f, --force:** If an existing destination file cannot be opened, remove it and try again.
- **-i, --interactive:** Prompt whether to overwrite an existing destination file.
- **-n, --no-clobber:** Do not overwrite an existing file.
- **-u, --update:** Move only when the source file is newer than the destination file or when the destination file is missing.
- **-v, --verbose:** Print the name of each file before moving it.

```
mv
```

## piping |

**Description:** piping is a command-line utility that connects the output of one command to the input of another command.

**example:**

- **cat file.txt | grep 'a':** search for the letter 'a' in the file.



- `ls -l | grep 'file'`: search for files in the current directory.
- `ls -l | grep 'file' | wc -l`: count the number of files in the current directory.

```
command1 | command2
```

## ps

**Description:** ps is a command-line utility that displays information about processes.

### Options:

- `-a`: Display information about other users' processes.
- `-A`: Display information about all processes.
- `-c`: Display the command name.
- `-e`: Display information about all processes.
- `-f`: Display full-format listing.
- `-H`: Display a tree of processes.
- `-l`: Display long format.
- `-p`: Display information about the specified process IDs.
- `-r`: Display processes in reverse order.
- `-t`: Display information about the specified terminal.
- `-u`: Display information about the specified user.
- `-x`: Display information about processes without controlling terminals.

```
ps
```

## pwd

**Description:** pwd is a command-line utility that prints the current working directory.

### Options:

- `-L`: Print the logical current working directory.
- `-P`: Print the physical current working directory.

```
pwd
```

## read

**Description:** read is a command-line utility that reads a line from the standard input.

### Options:

- `-p`: Display the prompt.
- `-r`: Do not allow backslashes to escape any characters.
- `-s`: Do not echo input coming from a terminal.
- `-t`: Time out and return failure if a complete line of input is not read within n seconds.
- `-u`: Read input from the specified file descriptor.

```
read
```

## redirecting >

**Description:** redirecting is a command-line utility that redirects the output of a command to a file.

**example:**

- `ls -l > file.txt`: save the output of the command to a file.
- `echo 'Hello, World!' > file.txt`: save the text to a file.
- `cat file.txt > file2.txt`: copy the content of the file to another file.

```
command > file.txt
```

## redirecting >>

**Description:** redirecting is a command-line utility that appends the output of a command to a file.

**example:**

- `ls -l >> file.txt`: append the output of the command to a file.
- `echo 'Hello, World!' >> file.txt`: append the text to a file.
- `cat file.txt >> file2.txt`: append the content of the file to another file.

```
command >> file.txt
```

## redirecting <

**Description:** redirecting is a command-line utility that redirects the input of a command from a file.

**example:**

- `grep 'a' < file.txt`: search for the letter 'a' in the file.
- `wc -l < file.txt`: count the number of lines in the file.

```
command < file.txt
```

## redirecting <<

**Description:** redirecting is a command-line utility that redirects the input of a command from a string.

**example:**

- `grep 'a' << 'Hello, World!'`: search for the letter 'a' in the string.
- `wc -l << 'Hello, World!'`: count the number of lines in the string.

```
command << 'string'
```

## redirecting < >

**Description:** redirecting is a command-line utility that redirects the input of a command from a file and the output of the command to another file.

**example:**

- `grep 'a' < file.txt > output.txt`: search for the letter 'a' in the file and save the output to a file.
- `wc -l < file.txt > output.txt`: count the number of lines in the file and save the output to a file.

```
command < file.txt > output.txt
```

## redirecting < >>

**Description:** redirecting is a command-line utility that redirects the input of a command from a file and appends the output of the command to another file.

**example:**

- `grep 'a' < file.txt >> output.txt`: search for the letter 'a' in the file and append the output to a file.
- `wc -l < file.txt >> output.txt`: count the number of lines in the file and append the output to a file.

```
command < file.txt >> output.txt
```

## redirecting << >

**Description:** redirecting is a command-line utility that redirects the input of a command from a string and the output of the command to a file.

**example:**

- `grep 'a' << 'Hello, World!' > output.txt`: search for the letter 'a' in the string and save the output to a file.
- `wc -l << 'Hello, World!' > output.txt`: count the number of lines in the string and save the output to a file.

```
command << 'string' > output.txt
```

## redirecting << >>

**Description:** redirecting is a command-line utility that redirects the input of a command from a string and appends the output of the command to a file.

**example:**

- `grep 'a' << 'Hello, World!' >> output.txt`: search for the letter 'a' in the string and append the output to a file.
- `wc -l << 'Hello, World!' >> output.txt`: count the number of lines in the string and append the output to a file.

```
command << 'string' >> output.txt
```

## redirecting 2>

**Description:** redirecting is a command-line utility that redirects the error output of a command to a file.

**example file output:**

- `/dev/null`: discard the output.
- `error.txt`: save the output to a file.
- `2>&1`: redirect the error output to the standard output.
- `/dev/tty`: write the output to the terminal.

**example:**

- `ls -l 2> error.txt`: save the error output of the command to a file.
- `echo 'Hello, World!' 2> error.txt`: save the error text to a file.
- `cat file.txt 2> error.txt`: copy the content of the file to another file.

```
command 2> error.txt
```

## redirecting 2>&1

**Description:** redirecting is a command-line utility that redirects the error output of a command to the standard output.

**example:**

- `ls -l 2>&1`: redirect the error output of the command to the standard output.
- `echo 'Hello, World!' 2>&1`: redirect the error text to the standard output.
- `cat file.txt 2>&1`: copy the content of the file to the standard output.

```
command 2>&1
```

## redirecting 1>&2

**Description:** redirecting is a command-line utility that redirects the standard output of a command to the error output.

**example:**

- `ls -l 1>&2`: redirect the standard output of the command to the error output.
- `echo 'Hello, World!' 1>&2`: redirect the text to the error output.
- `cat file.txt 1>&2`: copy the content of the file to the error output.

```
command 1>&2
```

## redirecting &>

**Description:** redirecting is a command-line utility that redirects the standard output and error output of a command to a file.

### example file output:

- `/dev/null`: discard the output.
- `error.txt`: save the output to a file.
- `2>&1`: redirect the error output to the standard output.
- `/dev/tty`: write the output to the terminal.

### example:

- `ls -l &> output.txt`: save the output of the command to a file.
- `echo 'Hello, World!' &> output.txt`: save the text to a file.
- `cat file.txt &> output.txt`: copy the content of the file to another file.

```
command &> output.txt
```

## rev

**Description:** rev is a command-line utility that reverses the characters in each line of a file.

```
rev
```

## rm

**Description:** rm is a command-line utility that removes files and directories.

### Options:

- `-f, --force`: Ignore nonexistent files and do not prompt.
- `-i`: Prompt before every removal.
- `-r, --recursive`: Remove directories and their contents recursively.
- `-v, --verbose`: Explain what is being done.
- `-d, --dir`: Remove empty directories.
- `-I`: Prompt once before removing more than three files or when removing recursively.
- `-P`: Prompt before every removal.
- `-R`: Remove directories and their contents recursively.
- `-v`: Explain what is being done.

```
rm
```

## rmdir

**Description:** rmdir is a command-line utility that removes empty directories.

### Options:

- `-p, --parents`: Remove parent directories if they are empty.
- `-v, --verbose`: Print a message for each removed directory.

```
rmdir
```

## sh

**Description:** sh is a command-line utility that runs the Bourne shell.

### Options:

- **-c**: Read commands from the string argument.
- **-i**: Start an interactive shell session.
- **-s**: Read commands from the standard input.
- **-D**: A list of all double-quoted strings preceded by \$ is printed on the standard output.
- **-x**: Print commands and their arguments as they are executed.

```
sh
```

## sort

**Description:** sort is a command-line utility that sorts lines of text files.

### Options:

- **-b, --ignore-leading-blanks**: Ignore leading blanks.
- **-d, --dictionary-order**: Consider only blanks and alphanumeric characters.
- **-f, --ignore-case**: Fold lower case to upper case characters.
- **-g, --general-numeric-sort**: Compare according to general numerical value.
- **-i, --ignore-nonprinting**: Ignore nonprinting characters.
- **-M, --month-sort**: Compare (unknown) < 'JAN' < ... < 'DEC'.
- **-n, --numeric-sort**: Compare according to string numerical value.
- **-r, --reverse**: Reverse the result of comparisons.
- **-R, --random-sort**: Sort by random hash of keys.
- **-V, --version-sort**: Natural sort of (version) numbers within text.
- **-c, --check**: Check whether the input is sorted.
- **-o, --output**: Write result to the specified file.
- **-s, --stable**: Stabilize sort by disabling last-resort comparison.
- **-t, --field-separator**: Use the specified field separator.
- **-k, --key**: Sort via a key; KEYDEF gives location and type.
- **-m, --merge**: Merge already sorted files.
- **-u, --unique**: Suppress lines that are repeated.
- **-z, --zero-terminated**: End lines with 0 byte, not newline.

```
sort
```

## tail

**Description:** tail is a command-line utility that displays the last N lines of a file.

### Options:

- **-n, --lines**: Print the last N lines of each file.
- **-f, --follow**: Output appended data as the file grows.

- **-F**: Same as **--follow=name** **--retry**.
- **-c**, **--bytes**: Print the last N bytes of each file.
- **-q**, **--quiet**, **--silent**: Never output headers giving file names.
- **-v**, **--verbose**: Always output headers giving file names.
- **--pid**: With **-f**, terminate after the process with the given PID dies.
- **--max-unchanged-stats**: With **--follow=name**, reopen a FILE which has not changed size after N (default 5) iterations to see if it has been unlinked or renamed (this is the usual case of rotated log files). With **inotify**, this option is rarely useful.

#### examples:

- **tail -n 5 file.txt**: print the last 5 lines of the file.
- **tail -n +5 file.txt**: print all lines starting from the 5th line of the file.
- **tail -n -5 file.txt**: print all lines except the last 5 lines of the file.

```
tail
```

## touch

**Description:** touch is a command-line utility that changes file timestamps. If the file does not exist, it creates an empty file.

#### Options:

- **-a**: Change the access time.
- **-c**: Do not create the file if it does not exist.
- **-d**: Use the specified time.
- **-m**: Change the modification time.
- **-r**: Use the timestamp of the specified file.
- **-t**: Use the specified time.

#### example:

- **touch file.txt**: create an empty file.
- **touch -d '2022-01-01 12:00:00' file.txt**: change the timestamp of the file.
- **touch -r file.txt file2.txt**: change the timestamp of the file to another file.

```
touch
```

## WC

**Description:** wc is a command-line utility that counts the number of lines, words, and characters in a file.

#### Options:

- **-c**, **--bytes**: Print the byte counts.
- **-m**, **--chars**: Print the character counts.
- **-l**, **--lines**: Print the newline counts.
- **-L**, **--max-line-length**: Print the length of the longest line.
- **-w**, **--words**: Print the word counts.

## whereis

**Description:** whereis is a command-line utility that locates the binary, source, and manual page files for a command.

**Options:**

- **-b**: Search for binaries.
- **-m**: Search for manual pages.
- **-s**: Search for sources.

```
whereis
```

## which

**Description:** which is a command-line utility that locates the binary files of a command.

**Options:**

- **-a**: Print all matching executables in PATH, not just the first.
- **-s**: No output, just return 0 if any of the executables are found, or 1 if none are found.
- **-v**: Print a more verbose output.

```
which
```